A black background with red lines

Description automatically generated

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**SPECIALIZAREA CALCULATOARE**

**PROIECT SI**

**Vehicle Type Recognition**

Disciplina : Sisteme Inteligente

Student : Pojar Andrei-Gabriel

An : III

Grupa : 30233

Profesor coordonator : Varvara Raluca

Am preluat un set de date de clasificare a imaginilor de vehicule care conține imagini a patru tipuri diferite de vehicule: mașină, camion, autobuz și motocicletă. Setul de date este creat pentru a ajuta să se dezvolte și să evalueze modele de clasificare a imaginilor pentru identificarea diferitelor tipuri de vehicule din imagini. Acest set de date a fost preluat de pe Kaggle: <https://www.kaggle.com/datasets/kaggleashwin/vehicle-type-recognition/data>

A screenshot of a computer

Description automatically generated

1. **Montarea unității Google Drive, definirea căilor și dezarhivarea fișierului care conține setul de date**

**A close-up of a computer screen

Description automatically generated**

1. **Definirea unei clase care se ocupă de încărcarea datelor din director, definirea unui obiect pe care se aplică o serie de transformări asupra imaginilor(augmentări)**

Aici definesc mai întâi o serie de augmentări care ulterior vor fi aplicate pe imaginile dintr-un lot.

Apo fac instanțierea setului de date și a dataloader-ului(pt a încărca datele de antrenare în loturi). Fiecare lot va avea 32 de imagini( de la setarea parametrului batch size la 32) și apoi se extrage un lot de antrenare și se afișeaza primele 4 imagini din acesta.

**A red bus and a motorcycle

Description automatically generated**

1. **Preprocesare, histograma și distribuția claselor în setul de date**

Mai întâi de toate creez un dataset nou prin aplicarea unor transformări de preprocesare. Apoi vom calcula media și deviația standard pentru setul de date preprocesat.

Aici afișez histograma pentru distribuția valorilor pixelilor în imagini, respectiv distribuția claselor în setul de date. Putem observa că fiecare clasă are același număr de imagini, deci toate clasele sunt egale ca număr de imagini.

**A screenshot of a computer

Description automatically generated**

1. **Model de recunoaștere a vehiculelor bazat pe o rețea neuronală convoluțională cu ajutorul arhitecturii ResNet**

Aici vom incepe cu inițializarea unui backbone de retea neuronală folosind arhitectura ResNet-18 preantrenată disponibilă în torchvision(acesta va fi utilizat pentru extragerea caracteristicilor din imagini).

Aici outputul este defapt arhitectura completă a modelului de recunoaștere a vehiculelor , inclusiv backbone-ul ResNet cu modificările aduse la stratul complet conectat pentru a se potrivi cu numărul de clase specificat.

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer program

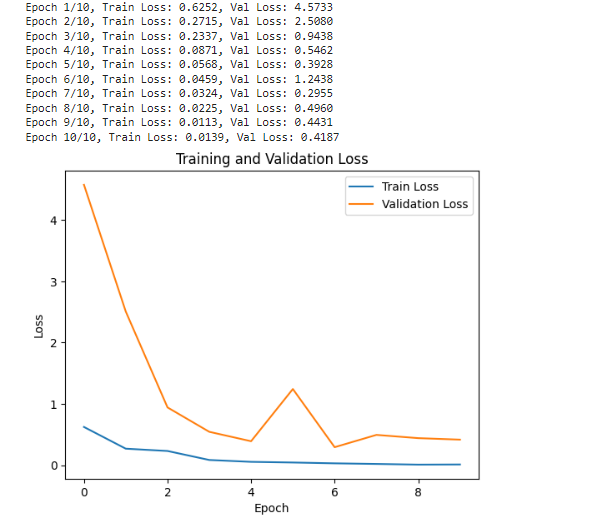
Description automatically generated**

1. **Antrenare și calculare train loss și validation loss pentru modelul bazat pe arhitectura ResNet**

Aici am împărțit setul de date în antrenare și validare și am calculat în timpul antrenării și validării funcția de pierdere și am afișat-o într-un grafic pentru a putea vedea cum evoluează.

Aceste valori și grafic le-am obținut folosind funcția de loss CrossEntropyLoss și optimizerul Adam la care i-am setat parametrul de lr la 0.001.

**Batch size-ul e setat aici la 32**

****

**Batch size-ul e setat aici la 8**

A graph of loss and validation

Description automatically generated with medium confidence

**Batch size-ul e setat aici la 64**

**A graph with numbers and lines

Description automatically generated**

Observații: Batch size=8 - Antrenament mai îndelungat, dar poate duce la o mai bună convergență și la o precizie mai mare pe setul de date de validare.

Batch size=32 - Oferă un echilibru între stabilitatea actualizărilor ponderilor și viteza de antrenament.

Batch size=64 - Antrenament mai rapid, dar poate duce la o convergență mai puțin precisă din cauza estimărilor zgomotoase ale gradientului.

1. **Antrenare și calculare train loss și validation loss cu model bazat pe arhitectura ResNet + Relu**

**Batch size-ul e setat aici la 32**

Aceste valori și grafic le-am obținut folosind funcția de loss CrossEntropyLoss și optimizerul Adam la care i-am setat parametrul de lr la 0.001.

A graph with numbers and a line

Description automatically generated

1. **Experimentarea cu learning rate folosind un scheduler**

Aici am folosit un scheduler pentru learning rate și optimizerul Adam la care i-am setat parametrul de lr la 0.001.

A graph with numbers and a line

Description automatically generated

Aici am folosit un scheduler pentru learning rate și optimizerul Adam la care i-am setat parametrul de lr la 0.01.

A graph with numbers and a line

Description automatically generated

1. **Funcții de pierdere( Cross entropy, Kullback-Leibler Divergence Loss)**

Am calculat cele două funcții de pierdere cu ajutorul unor funcții din torchvision. După aceea voi defini o funcție de antrenare care e folosită pentru testarea celor două funcții de pierdere. Ca optimizator am folosit Adam și i-am setat un lr de 0.001, am declarat 10 epoci pentru testarea funcțiilor și ca model am ales modelul definit cu ajutorul arhitecturii ResNet preantrenat.

Aici primul set de valori a celor 10 epoci sunt pentru funcția de pierdere Cross Entropy, iar al doilea pentru Kullback-Leibler Divergence Loss.

A screenshot of a computer

Description automatically generated

Putem observa ca dacă folosim ca funcție de pierdere KL Divergence loss-ul atât pentru train cât și pentru validation este aproape de 0.

A graph of a graph with text

Description automatically generated

1. **Acuratețe, recall, matrice de confuzie**

Aici am calculate accuracy care se referă la exactitatea de previziuni corecte între etichetele reale și cele prezise, recall-ul reprezintă sensibilitatea modelului, iar apoi am afișat matricea de confuzie care oferă o imagine a performanței modelului prin compararea etichetelor reale și a celor prezise.

A screenshot of a graph

Description automatically generated

1. **Experimentare cu optimizatorii: SGD, Adam, Rmsprop pentru modelul cu ResNet**

Learning rate=0.001, batch size pentru model=32

**Experimentare cu SGD:**

Pierderile de antrenare și de validare par să scadă treptat odată cu epocile, ceea ce indică o convergență stabilă.

A screenshot of a computer

Description automatically generated

**Experimentare cu Adam:**

Pierderile de antrenare și de validare par să scadă inițial, dar apoi să oscileze în timpul antrenării.

A screenshot of a computer

Description automatically generated

**Experimentare cu RMSprop:**

Pierderile de antrenare și de validare prezintă o fluctuație semnificativă, iar acuratețea finală este mult mai mica. Acest comportament poate indica o instabilitate în convergență sau poate fi cauzat de o rată de învățare inadecvată.

A screenshot of a computer

Description automatically generated

**Rezultate finale:**

SGD și Adam au prezentat performanțe relativ similare, cu o acuratețe finală apropiată. RMSprop a avut o performanță semnificativ mai slabă în acest caz. SGD pare să fie cel mai stabil, având o convergență mai lină și mai constantă în comparație cu Adam și RMSprop. Adam a prezentat o convergență mai fluctuantă, iar RMSprop a avut probleme semnificative de convergență în acest experiment.

A screenshot of a computer

Description automatically generated

**Graficele rezultate:**

**A graph of a line and a line

Description automatically generated**

**A graph of a person with a red line

Description automatically generated**